Probabilistic Model-based Imitation Learning

Peter Englert[1], Alexandros Paraschos[1], Jan Peters[1,2], and Marc Peter Deisenroth[1]

[1]Department of Computer Science, Technische Universität Darmstadt, Germany.
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany.
{englert|paraschos|peters|marc}@ias.tu-darmstadt.de

Abstract

Efficient skill acquisition is crucial for creating versatile robots. One intuitive way to teach a robot new tricks is to demonstrate a task and enable the robot to imitate the demonstrated behavior. This approach is known as imitation learning. Classical methods of imitation learning, such as inverse reinforcement learning or behavioral cloning, suffer substantially from the correspondence problem when the actions (i.e., motor commands, torques or forces) of the teacher are not observed or the body of the teacher differs substantially, e.g., in the actuation. To address these drawbacks we propose to learn a robot-specific controller that directly matches robot trajectories with observed ones. We present a novel and robust probabilistic model-based approach for solving a probabilistic trajectory matching problem via policy search. For this purpose, we propose to learn a probabilistic model of the system, which we exploit for mental rehearsal of the current controller by making predictions about future trajectories. These internal simulations allow for learning a controller without permanently interacting with the real system, which results in a reduced overall interaction time. Using long-term predictions from this learned model, we train robot-specific controllers that reproduce the expert's distribution of demonstrations without the need to observe motor commands during the demonstration. The strength of our approach is that it addresses the correspondence problem in a principled way. Our method achieves a higher learning speed than both model-based imitation learning based on dynamics motor primitives and trial-and-error based learning systems with hand-crafted cost functions.We successfully applied our approach to imitating human behavior using a tendon-driven compliant robotic arm. Moreover, we demonstrate the generalization ability of our approach in a multi-task learning set-up.

1 Introduction

Programming robots to perform complex tasks is difficult with classical methods for instructing robots, such as textual or GUI-driven programming techniques [8]. These methods require a large amount of work for programming a single task, and transfer to new environments is often not straightforward. Especially for programming versatile robots, where fast learning of new tasks in changing environments is necessary, these programming methods are often impractical.

Imitation learning (IL) is an approach to address such skill acquisition problems in an elegant way: A teacher's demonstration of a task is recorded, and, subsequently, learning algorithms transfer the task to a robot [2,4]. Especially for tasks that humans can perform well, this approach is often more straightforward for transferring skills than programming methods. Another advantage is that if robot movements resemble human movements, it is more likely that they will be accepted by humans, which, from a psychological point of view, is desirable when integrating robots into domestic environments.

Common IL methods include behavioral cloning [6] and inverse reinforcement learning [29]. In behavioral cloning, demonstrated trajectories are used to learn a policy mapping from observed states to controls. Subsequently, the robot applies the policy. In inverse reinforcement learning, the demonstrations are used to learn the teacher's cost function. Subsequently, a policy is learned that minimizes the learned cost function.

One key challenge in imitation learning is the *correspondence problem* [25]: If the body of the teacher and the robot differ, finding an adequate mapping from the teacher's demonstrations to the robot is non-trivial. The

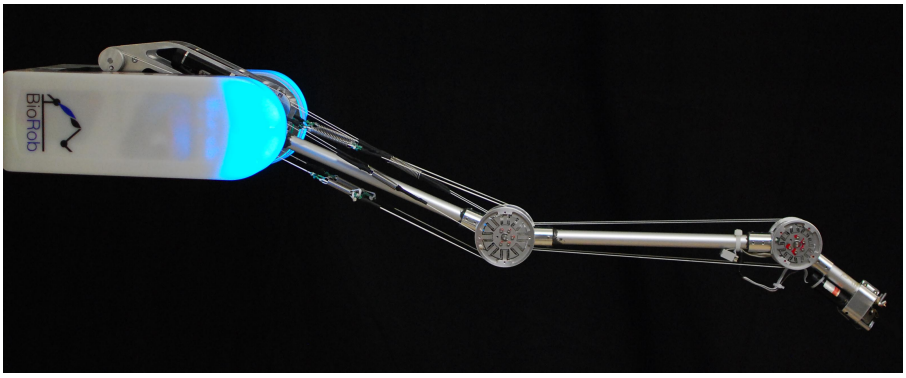


Figure 1: The BioRob^{TM} is a compliant, biomechanically-inspired robot manipulator with drive cables and springs, which represent tendons and their elasticity. Classical control approaches based on rigid body dynamics are unrealistic for this robot because they ignore the cable-driven properties and the elasticity of the tendons.

correspondence problem can occur in many different forms: One form is due to different anatomies between the teacher and the robot. As a consequence, some demonstrated behaviors of the teacher may not be executable by the robot. Another form of the correspondence problem are different dynamics properties between teacher and robot. For example, robots often have torque limits, such that the robot's joints cannot achieve the same velocities as the ones of the teacher.

In this paper, we propose a novel model-based imitation learning approach that addresses the correspondence problem and allows robots to efficiently acquire new behaviors from multiple expert demonstrations. The key idea is to directly match the predicted state trajectory of the robot with the teacher's demonstrations by learning a robot-specific controller. This gives us the advantage that it is not necessary to record the actions of the expert demonstrations, which allows us to choose from a wider range of demonstration methods (e.g., visual motion recordings, recordings with other robots) and to use the same demonstrations for teaching multiple different robots.

Our approach exploits a forward model of the robot's dynamics to generate trajectory predictions for a given controller. Using these simulations instead of sampling real robot trajectories reduces the interaction time with the robot. Such data-efficient learning saves experimental time and reduces the number of repairs. In the absence of a good parametric robot model, we propose to learn a data-driven forward model using a probabilistic non-parametric Gaussian process [33]. Learning a non-parametric forward model is especially suited for robots, where it is difficult to model the robot's dynamics with classical approaches like rigid-body dynamics (e.g., the BioRob^{TM}, see Figure 1). A probabilistic model allows us to take uncertainty about the robot's dynamics into account, which reduces learning bias due to model errors [3, 13, 37], a problem that is particularly pronounced when only a few samples and no informative prior knowledge is available. Furthermore, we do not need to make potentially unrealistic assumptions (e.g., about rigid-body dynamics or linear friction), which are typically made when learning parametric forward models in robotics [39].

The contribution of this paper is a model-based imitation learning framework based on probabilistic trajectory matching that addresses the correspondence problem in a principled way. We show that our IL approach learns faster and more robust than related approaches and demonstrate its generalization properties on an elastic compliant robot manipulator, see Figure 1.

The rest of the paper is structured as follows: In Section 2, we present related work on skill learning in robotics. In Section 3, we formulate the problem set-up and provide some background on Gaussian processes. In Section 4, we describe our model-based imitation learning approach, where we use reinforcement learning methods to find optimal policies. In Section 5, we demonstrate the viability of our approach on both simulated and real robots, such as the biomechanically-inspired manipulator shown in Figure 1.

2 Related Work

Research in the field of imitation learning devised techniques that differ in the way the demonstrations are provided (e.g., motion capture [41], physical interaction [7]), the level at which the imitation happens (e.g., at the symbolic [43] or trajectory level [10]), whether they use a system model, and whether/how they employ cost functions for the task. A detailed survey about learning skills from demonstration can be found in [2].

A classic form of imitation learning is Behavioral Cloning (BC). In BC, the behavior of a skilled human is recorded and, subsequently, an induction algorithm is executed over the traces of the behavior [6]. In classical BC, the objective of cloning observed expert demonstrations is phrased as a supervised learning problem. This problem is solved by learning a function from demonstration data that maps states to actions, i.e., a policy. An impressive early application of BC was the autonomous vehicle ALVINN [31], which learned a neural-network policy for driving a car using recorded state-action training pairs of a human driver. Advantages of BC are the straightforward application and the clean-up effect, i.e., the smoothing of noisy imperfect demonstrations. However, BC restricts the ways how the demonstrations can be recorded because it also needs to observe the action signals during the demonstration. Additionally, it suffers severely from the correspondence problem because it directly maps recored states to recorded actions without taking the anatomy and physics of the robot into account. Therefore, BC is not robust to changes in either the control task or the environment and cannot provide strong performance guarantees.

One more recent approach of imitation learning is the use of Dynamic Movement Primitives (DMP) [35, 36]. DMPs are a representation of movements as nonlinear differential equations that provide the option to modify the movement in different ways, e.g., goal position or duration. The shape of DMPs is learned from demonstrated trajectories. Since DMPs are typically used in the robot's joint space, demonstrations in task space require an additional mapping. Multiple extensions and modification for DMPs exist, e.g., real-time goal adaption [17] or generalizing movements with a mixture of DMPs [23].

Inverse Reinforcement Learning (IRL) is a form of imitation learning that automatically extracts a cost function from demonstrations of a task [9,29]. Subsequently, a policy is found that minimizes this cost function. By minimizing the teacher's cost function, the learned policy is supposed to produce a behavior similar to the demonstration. Hence, IRL is suited for tasks where the hand-crafted definition of a suitable cost function is difficult (e.g., parking lot navigation [1] or quadruped locomotion [20]). Drawbacks of IRL are that the performance of most methods rely on feature selection, which can strongly bias the performance.

Transferring a skill through imitation learning limits the performance of the robot to the skill of the teacher that provided the demonstration. Reinforcement Learning (RL) is a common technique to improve skills after applying imitation learning. RL [40] is an approach, where a task-specific cost function is minimized. RL has been successfully used in robotics applications for learning the ball-in-a-cup game [19] and helicopter hovering [28]. However, a major difficulty in RL is engineering a suitable cost function for more complex tasks.

In this paper, we phrase imitation learning directly as a probabilistic trajectory-matching problem. We learn policies that maximize the similarity between distributions over demonstrated expert trajectories and predicted robot trajectories. As a difference measure between trajectory distributions we use the Kullback-Leibler (KL) divergence. We show that our imitation learning problem is equivalent to a reinforcement learning problem with an induced cost function. Compared to other IL algorithms our approach addresses the correspondence problem in two ways: First, robot-specific controllers are learned that explicitly take the robot's torque limits and anatomic differences between the robot and the teacher into account. Second, it is not necessary to record actions of the teacher's demonstrations.

3 Problem Statement and Background

Throughout this paper, we use the following notation: We denote states by $\boldsymbol{x} \in \mathbb{R}^D$ and actions by $\boldsymbol{u} \in \mathbb{R}^E$, respectively. Furthermore, we define a trajectory $\boldsymbol{\tau}$ as a sequence of states $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ for a fixed finite time horizon T. Our goal is to learn a policy π such that the robot's trajectory match demonstrated trajectories. We assume a parametrized state-feedback policy π such that $\boldsymbol{u} = \pi(\boldsymbol{x}, \boldsymbol{\theta})$ with policy parameters $\boldsymbol{\theta}$.

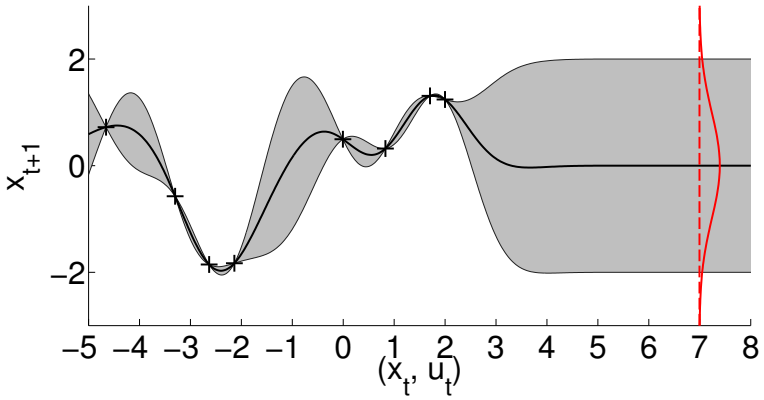


Figure 2: Learning a probabilistic forward model with Gaussian processes. The horizontal axis represents state-action input pairs (x_t, u_t) of the model and the vertical axis represents the predicted next state x_{t+1}. The black crosses denote the training data and the grey shaded area represents two times the standard deviation of the predictive uncertainty. It can be seen, that in the region around the training points, the predictions are more certain than at inputs further away. The red line denotes a test input in a region where we have not observed any data. Here, the model would return the prior with our initial uncertainty.

3.1 Problem Statement

We assume that the teacher provides n trajectories $\boldsymbol{\tau}_i$ from which the robot should learn to imitate the demonstrated behavior. We assume that the demonstrated trajectories start from an initial state distribution $p(\boldsymbol{x}_0)$ to account for variability in the demonstrations. Since we have n different trajectories, we use the probability distribution $p(\boldsymbol{\tau}^{\text{exp}})$ to represent the variability of the demonstrated trajectories. For example, a transporting task requires at the pick-up position of the object a higher accuracy, and, hence, the variance of the demonstrations is smaller there than during the transport itself. We also use a probability distribution for representing the robot trajectory predictions $p(\boldsymbol{\tau}^{\pi})$, which allows us to represent uncertainty of the robot's dynamics in a principled way.

Our goal is to find a policy π, such that the robot's behavior matches the demonstrations. For this purpose, our objective is to match the distribution over predicted trajectories $p(\boldsymbol{\tau}^{\pi})$ with the distribution over demonstrated trajectories $p(\boldsymbol{\tau}^{\text{exp}})$. As a similarity measure between these distributions, we use the KL divergence [38]. Hence, our imitation learning objective is to find a policy such that

$$\pi^* \in \arg\min_{\pi} \text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}}) || p(\boldsymbol{\tau}^{\pi})\big)\,. \tag{1}$$

For predicting $p(\boldsymbol{\tau}^{\pi})$, we exploit a learned model of the robot's forward dynamics, which we describe in the following.

3.2 Learning Probabilistic Forward Models

We learn a forward model of the robot's dynamics for performing internal simulations, which is related to the concept that humans rely on internal models for planning, control and learning of their dynamics behavior [42]. A forward model f maps a state $\boldsymbol{x}_t$ and action $\boldsymbol{u}_t$ of the system to the next state $\boldsymbol{x}_{t+1}$. In our case, we assume that $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\epsilon)$ is i.i.d. Gaussian noise with $\boldsymbol{\Sigma}_\epsilon = \text{diag}([\sigma_1^2 \dots \sigma_D^2])$. Such a model represents the transition dynamics of a robot. We represent the model by a Gaussian Process (GP), i.e., a probability distribution over models [33]. A GP is defined as a collection of random variables, any finite number of which is Gaussian distributed.

Since a GP is a consistent, non-parametric method, we do not have to specify a restrictive parametric model. The GP infers a posterior distribution over the underlying function f directly from the data, while the uncertainty about this estimate is represented as well. As training inputs to the GP, we use state-action pairs $(\boldsymbol{x}_t^{(i)}, \boldsymbol{u}_t^{(i)})$ and as targets the corresponding successors $\boldsymbol{x}_{t+1}^{(i)}$ for $i = 1, \dots, n$. Such a GP represents one-step

transitions in the form

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{x}_{t+1}|\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) \tag{2}$$

$$\text{with} \quad \boldsymbol{\mu}_{t+1} = \mathbb{E}_f[f(\boldsymbol{x}_t, \boldsymbol{u}_t)] = m_f(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{3}$$

$$\boldsymbol{\Sigma}_{t+1} = \text{var}_f[f(\boldsymbol{x}_t, \boldsymbol{u}_t)] = \sigma_f^2(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{4}$$

where m_f is the mean and σ_f^2 the variance of f. An example of such a model is visualized in Figure 2, where the horizontal axis represents state-action input pairs $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ and the vertical axis represents the predicted next state $\boldsymbol{x}_{t+1}$.

A GP is completely specified by a mean function m and a covariance function k. The mean function allows to integrate prior knowledge about the underlying dynamics f (e.g., rigid-body dynamics) and the covariance function incorporates some high-level structured assumptions about the true underlying function (e.g., smoothness). We use an uninformative prior mean function $m \equiv 0$ for symmetry reasons and a squared exponential covariance function plus noise covariance

$$k(\tilde{\boldsymbol{x}}_p, \tilde{\boldsymbol{x}}_q) = \alpha^2 \exp\left(-\tfrac{1}{2}(\tilde{\boldsymbol{x}}_p - \tilde{\boldsymbol{x}}_q)^\top \boldsymbol{\Lambda}^{-1}(\tilde{\boldsymbol{x}}_p - \tilde{\boldsymbol{x}}_q)\right) + \delta_{pq}\boldsymbol{\sigma}_\epsilon^2 \tag{5}$$

with inputs of the form $\tilde{\boldsymbol{x}} = [\boldsymbol{x}^\top, \boldsymbol{u}^\top]^\top$, so that we obtain a smooth function. The parameter α^2 is the signal variance, $\boldsymbol{\Lambda} = \text{diag}([l_1^2, \ldots, l_D^2])$ is a matrix with the squared length-scales, and δ_{pq} is the Kronecker symbol, which is 1 when $p = q$, and 0 otherwise. The posterior predictive distribution at a test input $\tilde{\boldsymbol{x}}_\star$ is given by the mean and variance

$$m_f(\tilde{\boldsymbol{x}}_\star) = \boldsymbol{k}_\star^\top \boldsymbol{K}^{-1}\boldsymbol{y}, \tag{6}$$

$$\sigma_f^2(\tilde{\boldsymbol{x}}_\star) = \boldsymbol{k}_{\star\star} - \boldsymbol{k}_\star^\top \boldsymbol{K}^{-1}\boldsymbol{k}_\star \tag{7}$$

with $\boldsymbol{k}_\star := k(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{x}}_\star)$, $\boldsymbol{k}_{\star\star} := k(\tilde{\boldsymbol{x}}_\star, \tilde{\boldsymbol{x}}_\star)$, Gram matrix $\boldsymbol{K}$ with $K_{ij} = k(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$, and training inputs $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n]$ with corresponding targets $\boldsymbol{y} = [y_1, \ldots, y_n]^\top$. Equations (2)–(7) are used to simulate the system for a single time step and map the current state-action pair $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ onto a probability distribution over the next state $\boldsymbol{x}_{t+1}$ (see Figure 2). The GP model can be reused for different tasks as long the dynamics of the system do not change.

4 Probabilistic Imitation Learning via Trajectory Matching

Our goal is to imitate the expert's behavior by finding a policy π^* that minimizes the KL divergence between the distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over demonstrated trajectories and the distribution $p(\boldsymbol{\tau}^\pi)$ over predicted trajectories when executing a policy π, see Equation (1).

The KL divergence is a difference measure between two probability distributions and is defined for continuous distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ as

$$\text{KL}\big(p(\boldsymbol{x})||q(\boldsymbol{x})\big) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}. \tag{8}$$

The KL divergence has been widely used in the development of machine learning algorithms. In robotics research, it has been previously used to measure the distance between Hidden Markov Models [18, 24] and for mapping and localization algorithms [11].

4.1 Trajectory Representation

We approximate a distribution over trajectories $p(\boldsymbol{\tau}) = p(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T)$ by a Gaussian $\mathcal{N}(\boldsymbol{\tau}|\boldsymbol{\mu}_{\boldsymbol{\tau}}, \boldsymbol{\Sigma}_{\boldsymbol{\tau}})$ that factorizes according to

$$p(\boldsymbol{\tau}) \approx \prod_{t=1}^{T} p(\boldsymbol{x}_t) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)\,. \tag{9}$$

This simplifying assumption implies that $\boldsymbol{\Sigma}_{\boldsymbol{\tau}} \in \mathbb{R}^{TD \times TD}$ is block diagonal without cross-correlations among states at different time steps. In the following paragraphs, we describe how we compute the probability distributions over trajectories $p(\boldsymbol{\tau}^{\text{exp}})$ from demonstrations and $p(\boldsymbol{\tau}^{\pi})$ with model predictions for our objective in Equation (1).

4.1.1 Estimation of a Distribution over Expert Trajectories

The demonstrations of the teacher are converted such that they are time-aligned, i.e., each trajectory $\boldsymbol{\tau}_i$ consists of a sequence of T states. This can be achieved for example through dynamic time warping [34]. The mean and covariance matrix of the marginals $p(\boldsymbol{x}_t)$ are computed as unbiased estimates $p(\boldsymbol{x}_t) \approx \mathcal{N}(\hat{\boldsymbol{\mu}}_t^{\text{exp}}, \hat{\boldsymbol{\Sigma}}_t^{\text{exp}})$, where

$$\hat{\boldsymbol{\mu}}_t^{\text{exp}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_t^i \tag{10}$$

$$\hat{\boldsymbol{\Sigma}}_t^{\text{exp}} = \frac{1}{n-1}\sum_{i=1}^{n} (\boldsymbol{x}_t^i - \hat{\boldsymbol{\mu}}_t^{\text{exp}})(\boldsymbol{x}_t^i - \hat{\boldsymbol{\mu}}_t^{\text{exp}})^{\top}. \tag{11}$$

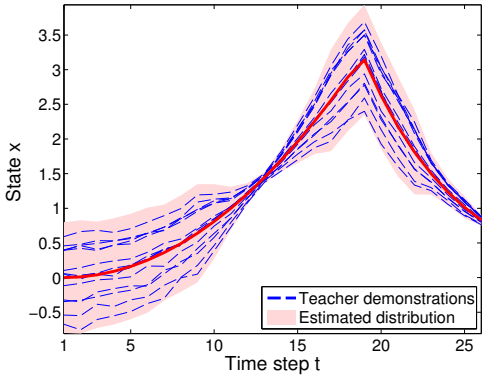


Figure 3: Estimation of a Gaussian distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over trajectories.

In Equations (10)–(11), $\boldsymbol{x}_t^i$ is the state after t time steps of the i^{th} demonstrated expert trajectory. This estimation yields an approximate Gaussian distribution over the expert trajectories

$$p(\boldsymbol{\tau}^{\text{exp}}) = \mathcal{N}(\hat{\boldsymbol{\mu}}^{\text{exp}}, \hat{\boldsymbol{\Sigma}}^{\text{exp}}) = \mathcal{N}\left(\begin{bmatrix} \hat{\boldsymbol{\mu}}_1^{\text{exp}} \\ \hat{\boldsymbol{\mu}}_2^{\text{exp}} \\ \vdots \\ \hat{\boldsymbol{\mu}}_T^{\text{exp}} \end{bmatrix}, \begin{bmatrix} \hat{\boldsymbol{\Sigma}}_1^{\text{exp}} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \hat{\boldsymbol{\Sigma}}_2^{\text{exp}} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \hat{\boldsymbol{\Sigma}}_T^{\text{exp}} \end{bmatrix} \right) \tag{12}$$

with a block diagonal covariance matrix $\hat{\boldsymbol{\Sigma}}^{\text{exp}}$. An illustration of such a trajectory representation is shown in Figure 3, where multiple teacher demonstrations (blue dashed lines) are estimated by a Gaussian (red shaded graph).

4.1.2 Predicting a Distribution over Robot Trajectories

We use the learned GP forward model described in Section 3.2 for iteratively predicting the state distributions $p(\boldsymbol{x}_1), \dots, p(\boldsymbol{x}_T)$ for a given policy π and an initial state distribution $p(\boldsymbol{x}_0)$. These long-term predictions are the marginal distributions of $p(\boldsymbol{\tau}^{\pi})$. Note that even for a given input pair $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_t, \boldsymbol{u}_t)$, the GP's prediction is a probability distribution, given by Equations (6)–(7). Iteratively computing the predictions $p(\boldsymbol{x}_1), \dots, p(\boldsymbol{x}_T)$, therefore, requires to predict with Gaussian processes at uncertain inputs [13, 32]. Computing $p(\boldsymbol{x}_{t+1})$ at an uncertain input $p(\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\tilde{\boldsymbol{x}}_t | \tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)$ requires integrating out both the uncertainty about the state-action pair $\tilde{\boldsymbol{x}}_t$ and the posterior uncertainty about the function $f \sim GP$ according to

$$p(\boldsymbol{x}_{t+1}) = \iint p(\boldsymbol{x}_{t+1} | \tilde{\boldsymbol{x}}_t) p(\tilde{\boldsymbol{x}}_t) df \, d\tilde{\boldsymbol{x}}_t, \tag{13}$$

where $\boldsymbol{x}_{t+1} = f(\tilde{\boldsymbol{x}}_t) = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$. The transition probability $p(\boldsymbol{x}_{t+1} | \tilde{\boldsymbol{x}}_t)$ is the posterior GP predictive distribution given in Equations (6)–(7), and $p(\tilde{\boldsymbol{x}}_t) = \mathcal{N}(\tilde{\boldsymbol{x}}_t | \tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)$ is assumed Gaussian. This mapping of an uncertain input $p(\boldsymbol{x}_t, \boldsymbol{u}_t)$ through a GP is visualized in Figure 4. The input distribution is shown in the bottom panel, and the GP is shown in the top left panel. The exact predictive distribution $p(\boldsymbol{x}_{t+1})$ from Equation (13)

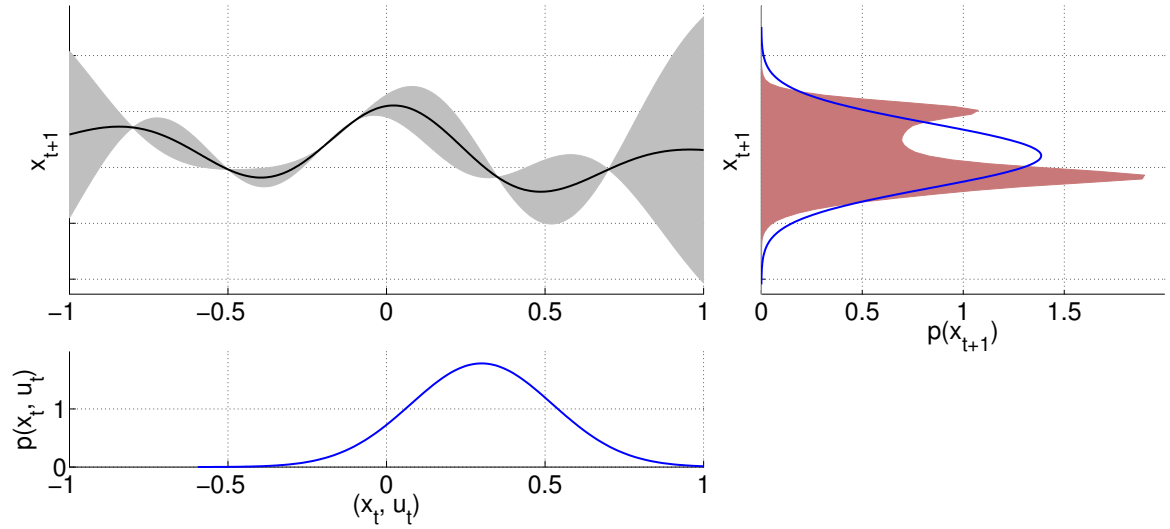


Figure 4: Visualization of predictions with Gaussian processes at uncertain inputs. The bottom panel shows the Gaussian input distribution $p(\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)$. The upper left panel shows the distribution $f \sim GP$. The upper right panel shows the exact prediction $p(\boldsymbol{x}_{t+1})$ shaded in red, which cannot be computed analytically (see Equation (13)). Therefore, we use exact moment matching to approximate this distribution with a Gaussian $\mathcal{N}(\boldsymbol{x}_{t+1}|\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$, which is drawn as a blue graph in the upper right panel.

is visualized in the top right panel of Figure 4 as the red shaded area. However, computing the exact distribution $p(\boldsymbol{x}_{t+1})$ given in Equation (13) is analytically intractable. Therefore, we approximate it by a Gaussian $p(\boldsymbol{x}_{t+1}) \approx \mathcal{N}(\boldsymbol{x}_{t+1}|\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$, which is shown as the blue graph in the top right panel of Figure 4.

For the Gaussian approximation of $p(\boldsymbol{x}_{t+1})$ we use exact moment matching. Following [13] the mean is computed (using the law of total expectation and Equation (6)) by

$$\begin{aligned}\boldsymbol{\mu}_{t+1} &= \mathbb{E}[f(\tilde{\boldsymbol{x}}_t)] = \mathbb{E}_{\tilde{x}}\big[\mathbb{E}_f[f(\tilde{\boldsymbol{x}}_t)|\tilde{\boldsymbol{x}}_t]\big] \overset{(3)}{=} \mathbb{E}_{\tilde{x}}\big[m_f(\tilde{\boldsymbol{x}}_t)\big] \\ &= \int m_f(\tilde{\boldsymbol{x}}_t)\mathcal{N}(\tilde{\boldsymbol{x}}_t|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)d\tilde{\boldsymbol{x}}_t \overset{(6)}{=} \int k(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{X}})\mathcal{N}(\tilde{\boldsymbol{x}}_t|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)d\tilde{\boldsymbol{x}}_t \boldsymbol{K}^{-1}\boldsymbol{y} = \boldsymbol{\beta}^\top \boldsymbol{q}\end{aligned} \tag{14}$$

with $\boldsymbol{\beta} = \boldsymbol{K}^{-1}\boldsymbol{y}$ and $\boldsymbol{q} = [q_1, \cdots, q_n]^\top$. Using Equation (5), the entries of $\boldsymbol{q}$ are given by

$$q_i = \int k(\tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_i)\mathcal{N}(\tilde{\boldsymbol{x}}_t|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)d\tilde{\boldsymbol{x}}_t = \alpha^2|\tilde{\boldsymbol{\Sigma}}_t\boldsymbol{\Lambda}^{-1} + \boldsymbol{I}|^{-\frac{1}{2}} \exp(-\tfrac{1}{2}\boldsymbol{\nu}_i^\top(\tilde{\boldsymbol{\Sigma}}_t + \boldsymbol{\Lambda})^{-1}\boldsymbol{\nu}_i), \tag{15}$$

where we used $\boldsymbol{\nu}_i := \tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{\mu}}_t$ with the training input $\tilde{\boldsymbol{x}}_i$. The predictive covariance matrix $\boldsymbol{\Sigma}_{t+1}$ can be derived similarly. The entries of the covariance matrix $\boldsymbol{\Sigma}_{t+1} \in \mathbb{R}^{D\times D}$ for the target dimension $a, b = 1, \ldots, D$ are

$$\sigma_{ab}^2 = \boldsymbol{\beta}_a^\top(\boldsymbol{Q} - \boldsymbol{q}_a\boldsymbol{q}_b^\top)\boldsymbol{\beta}_b + \delta_{ab}\left(\alpha_a^2 - \mathrm{tr}(\boldsymbol{K}^{-1}\boldsymbol{Q})\right) . \tag{16}$$

In Equation (16), the entries Q_{ij} of $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ are given by

$$\begin{aligned}Q_{ij} = \alpha_a^2\alpha_b^2\big|(\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1})\tilde{\boldsymbol{\Sigma}}_t + \boldsymbol{I}\big|^{-\frac{1}{2}} \exp\Big(-\tfrac{1}{2}(\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)^\top(\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b)^{-1}(\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)\Big) \\ \times \exp\Big(-\tfrac{1}{2}(\hat{\boldsymbol{z}}_{ij} - \tilde{\boldsymbol{\mu}}_t)^\top\big((\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1})^{-1} + \tilde{\boldsymbol{\Sigma}}_t\big)^{-1}(\hat{\boldsymbol{z}}_{ij} - \tilde{\boldsymbol{\mu}}_t)\Big)\end{aligned} \tag{17}$$

with

$$\hat{\boldsymbol{z}}_{ij} = \boldsymbol{\Lambda}_b(\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b)^{-1}\tilde{\boldsymbol{x}}_i + \boldsymbol{\Lambda}_a(\boldsymbol{\Lambda}_a + \boldsymbol{\Lambda}_b)^{-1}\tilde{\boldsymbol{x}}_j\,, \tag{18}$$

where $i, j = 1, \ldots, n$. For a detailed derivation of these results, see [13].

The GP predictions at uncertain inputs from Equations (14)–(18) allow the system to iteratively predict the long-term outcome for a policy π and a given distribution of the start state $\boldsymbol{x}_0$, which results in a probability distribution over trajectories $p(\boldsymbol{\tau}^\pi)$.

4.2 Natural Cost Function

In the previous sections, we detailed how to approximate $p(\boldsymbol{\tau}^{\text{exp}})$ and $p(\boldsymbol{\tau}^{\pi})$ by Gaussian distributions. For such a special case of two Gaussian distributions $p(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $q(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, the KL divergence in Equation (8) has the closed form expression

$$\text{KL}\left(p||q\right) = \tfrac{1}{2}\log\left|\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right| + \tfrac{1}{2}\text{tr}\left(\boldsymbol{\Sigma}_1^{-1}\left((\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^{\top} + \boldsymbol{\Sigma}_0 - \boldsymbol{\Sigma}_1\right)\right). \tag{19}$$

We use Equation (19) for our imitation learning approach as a cost function between probability distributions over trajectories $p(\boldsymbol{\tau}^{\text{exp}})$ and $p(\boldsymbol{\tau}^{\pi})$. The trajectory factorization in Equation (9) simplifies the KL divergence as it is suffices to sum up the individual KL divergences of the marginal distributions $p(\boldsymbol{x}_t^{\text{exp}})$ and $p(\boldsymbol{x}_t^{\pi})$, $t = 1, \ldots, T$, and we obtain the imitation learning objective function

$$J_{\text{IL}}^{\pi} = \text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}})||p(\boldsymbol{\tau}^{\pi})\big) = \sum_{t=1}^{T}\text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^{\pi})\big). \tag{20}$$

Here, we used the trajectory representations from Section 4.1 and Equation (8). Since the marginals $p(\boldsymbol{x}_t)$ are approximated by Gaussians, the KL divergence in Equation (20) can be evaluated in closed form by applying Equation (19).

Matching the distribution of the predicted state trajectory for a given policy with the distribution over expert trajectories by means of minimizing the KL divergence induces a natural cost function in a standard RL context [16]: Equation (20) shows that matching two factorized distributions by means of the KL divergence yields an additive objective function. Therefore, with $c(\boldsymbol{x}_t) = \text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^{\pi})\big)$, we can use RL methods to minimize Equation (20) since the IL objective J_{IL}^{π} corresponds to a RL long-term cost J_{RL}^{π} of the form

$$J_{\text{RL}}^{\pi} = \sum_{t=1}^{T} c(\boldsymbol{x}_t^{\pi}) = \sum_{t=1}^{T}\text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^{\pi})\big) = \sum_{t=1}^{T}\text{KL}\big(\mathcal{N}(\hat{\boldsymbol{\mu}}_t^{\text{exp}}, \hat{\boldsymbol{\Sigma}}_t^{\text{exp}})||\mathcal{N}(\boldsymbol{\mu}_t^{\pi}, \boldsymbol{\Sigma}_t^{\pi})\big) = J_{\text{IL}}^{\pi}. \tag{21}$$

In Equation (21), we used our assumption that trajectories are represented by Gaussian distributions with block-diagonal covariance matrices.

Since $\text{KL}\big(p(\boldsymbol{x}_t^{\text{exp}})||p(\boldsymbol{x}_t^{\pi})\big)$ corresponds to a RL long-term cost function, we can apply RL algorithms to find optimal policies. In principle, any algorithm that can predict trajectories of the current policy π is suitable. For instance, model-free methods based on sampling trajectories directly from the robot [30,40] or model-based RL algorithms that learn forward models of the robot and, subsequently, use them for predictions [5, 14, 15, 27] are suitable. In this paper, we exploit the relationship between RL and IL via trajectory matching and use a model-based policy search method with learned probabilistic forward models [13] to minimize the KL divergence $\text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}})||p(\boldsymbol{\tau}^{\pi})\big)$.

4.3 Policy Learning

For learning a policy that solves the imitation learning problem in Equation (1), we exploit the result in Equation (21) and use the PILCO (Probabilistic Inference for Learning COntrol) framework [13] as RL method for matching the trajectory distributions $p(\boldsymbol{\tau}^{\text{exp}})$ and $p(\boldsymbol{\tau}^{\pi})$. An overview of our method is given in Algorithm 1. Our objective is to find parameters $\boldsymbol{\theta}$ of a policy π that minimize the long-term cost in Equation (21). To find policy parameters $\boldsymbol{\theta}$, such that the distribution over the predicted trajectory matches the distribution over the expert trajectory, we minimize our cost function in Equation (20) by means of gradient-based optimization. The gradient of the IL objective J_{IL}^{π} with respect to the policy parameters $\boldsymbol{\theta}$ is

$$\frac{dJ_{\text{IL}}^{\pi}}{d\boldsymbol{\theta}} = \sum_{t=1}^{T}\left(\frac{\partial\text{KL}}{\partial\boldsymbol{\mu}_t^{\pi}}\frac{d\boldsymbol{\mu}_t^{\pi}}{d\boldsymbol{\theta}} + \frac{\partial\text{KL}}{\partial\boldsymbol{\Sigma}_t^{\pi}}\frac{d\boldsymbol{\Sigma}_t^{\pi}}{d\boldsymbol{\theta}}\right), \tag{22}$$

Algorithm 1 Probabilistic Model-based Imitation Learning

input: n expert trajectories $\boldsymbol{\tau}_i$ of a task
init: Estimate expert distribution over trajectories $p(\boldsymbol{\tau}^{\text{exp}})$ (see Section 4.1.1)
Record state-action pairs of the robot (e.g., through applying random control signals)
repeat
Learn/update probabilistic GP forward model (see Section 3.2)
Predict $p(\boldsymbol{\tau}^{\pi})$ (see Section 4.1.2)
Learn policy $\pi^{\star} \in \arg\min_{\pi} \text{KL}\big(p(\boldsymbol{\tau}^{\text{exp}}) || p(\boldsymbol{\tau}^{\pi})\big)$ (see Section 4.3)
Apply $\pi^{\star}$ to system and record data
until task learned

where we require the partial derivatives of the KL divergence with respect to the mean $\boldsymbol{\mu}_t^{\pi}$ and the covariance $\boldsymbol{\Sigma}_t^{\pi}$ of the predicted state distribution at time t. The corresponding partial derivatives are given by

$$\frac{\partial \text{KL}}{\partial \boldsymbol{\mu}_t^{\pi}} = -(\boldsymbol{\Sigma}_t^{\pi})^{-1}(\hat{\boldsymbol{\mu}}_t^{\text{exp}} - \boldsymbol{\mu}_t^{\pi}), \tag{23}$$

$$\frac{\partial \text{KL}}{\partial \boldsymbol{\Sigma}_t^{\pi}} = \tfrac{1}{2}(\boldsymbol{\Sigma}_t^{\pi})^{-1} - \tfrac{1}{2}(\boldsymbol{\Sigma}_t^{\pi})^{-1}\Big((\boldsymbol{\Sigma}_t^{\pi})^{-1} + (\hat{\boldsymbol{\mu}}_t^{\text{exp}} - \boldsymbol{\mu}_t^{\pi})(\hat{\boldsymbol{\mu}}_t^{\text{exp}} - \boldsymbol{\mu}_t^{\pi})^{\top}\Big)(\boldsymbol{\Sigma}_t^{\pi})^{-1}, \tag{24}$$

respectively. The derivatives of the mean $\boldsymbol{\mu}_t^{\pi}$ and covariance $\boldsymbol{\Sigma}_t^{\pi}$ with respect to $\boldsymbol{\theta}$ are the same as in [13]. All the derivatives in Equation (22) can be computed analytically and allow to use of fast gradient-based optimization methods, such as CG or BFGS.

With the KL divergence as difference measure between the estimated expert distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over trajectories and the predictive distribution $p(\boldsymbol{\tau}^{\pi})$ over trajectories, we have formulated model-based imitation learning as a reinforcement learning problem. Thereby, the KL divergence serves as an induced natural cost function. The analytic gradients of the loss function allow us to use gradient-based policy search methods. Therefore, we introduced all ingredients for performing probabilistic model-based imitation learning (see Algorithm 1) and solving the problem defined in Equation (1).

5 Experimental Results

In this section, we demonstrate the performance of our model-based imitation learning approach in different experiments. First, a policy for a swing-up task of a simulated double pendulum with two actuators was learned by means of imitation learning. Second, a tendon-driven real BioRob$^{\text{TM}}$ with complex internal dynamics learned to imitate demonstrations provided via kinesthetic teaching. We demonstrate that our imitation learning approach addresses the correspondence problem.

In the following experiments, we used a non-linear Radial Basis Function (RBF) network with axis-aligned Gaussian features ϕ as policy parametrization. This policy can be written as

$$\tilde{\pi}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{m} w_i \phi_i(\boldsymbol{x}) \quad \text{with} \tag{25}$$

$$\phi_i(\boldsymbol{x}) = \exp\big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_i)^{\top}\boldsymbol{\Gamma}^{-1}(\boldsymbol{x} - \boldsymbol{c}_i)\big), \tag{26}$$

weights w_i, centers $\boldsymbol{c}_i$, and widths γ_i of each dimension in $\boldsymbol{\Gamma} = \text{diag}(\gamma_1^2, \gamma_2^2, \ldots, \gamma_D^2)$. The parameters $\boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{c}, \boldsymbol{\gamma}\}$ of the RBF network were learned by minimizing J_{IL} in Equation (20) with gradient-based optimization methods. For taking the torque limits $u_{\max}$ appropriately into account during planning, we squash the policy $\tilde{\pi}$ through a sinusoidal function to obtain the torque-restricted policy

$$\pi(\boldsymbol{x}, \boldsymbol{\theta}) = u_{\max} \sin\big(\tilde{\pi}(\boldsymbol{x}, \boldsymbol{\theta})\big). \tag{27}$$

Therefore, the planning phase already takes the robot's torque limits into account, which restricts the policy to the mechanically admissible range.

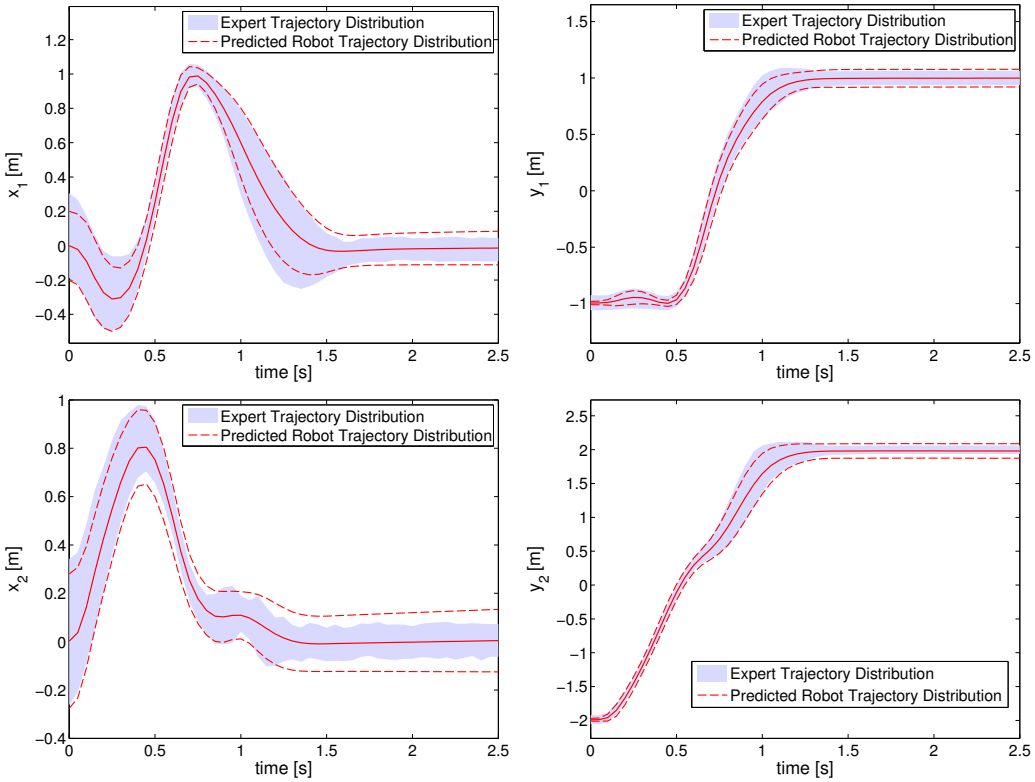


Figure 6: The Figures show the task space positions $[x_1, y_1, x_2, y_2]$ of the double pendulum. The blue shaded graph shows the expert trajectory distribution $p(\boldsymbol{\tau}^{\text{exp}})$ and the red graph shows the predicted robot trajectory distribution $p(\boldsymbol{\tau}^{\pi})$. In both cases, the twice the standard deviations are shown.

5.1 Double Pendulum

The double pendulum consists of two links and two actuated joints and was mounted on the ground (see Figure 5). The system state consisted of joint positions and velocities $\boldsymbol{x} = [q_1, q_1, \dot{q}_2, \dot{q}_2]^\top$; the motor torques served as actions $\boldsymbol{u} = [u_1, u_2]^\top$. The task was to swing up and balance the double pendulum in the inverted position, see Figure 5. Each link had a mass $m = 0.5\,\text{kg}$ and a length $l = 0.5\,\text{m}$. The motor torques were limited to the range $[-3.5, 3.5]\,\text{Nm}$. We used a sampling frequency of $10\,\text{Hz}$ at which the control signals could be changed and a total prediction horizon of $T = 2.5\,\text{s}$. For the RBF network in Equations (25)–(26) we used 100 basis functions, which resulted in 812 policy parameters $\boldsymbol{\theta}$.

The GP forward model was learned in joint space, but trajectory matching via minimizing the KL divergence was performed in task space, i.e., the observed trajectories consisted of a sequence of link positions $[x_1, y_1, x_2, y_2]$. Therefore, solving the imitation learning task required addressing one form of the correspondence problem. The distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over expert trajectories was based on five similar successful demonstrations of the task and created according to Section 4.1. Figure 6 shows the predicted robot trajectory distribution $p(\boldsymbol{\tau}^{\pi})$ in the blue shaded graph and the distribution over expert trajectories $p(\boldsymbol{\tau}^{\text{exp}})$, represented by the red graph. Both predictive distributions are shown in task space coordinates. The figure illustrates that the learned policy adapts to changing variability of the demonstrations, which is especially pronounced in the x_1-coordinate, i.e., the x-coordinate of the inner link.

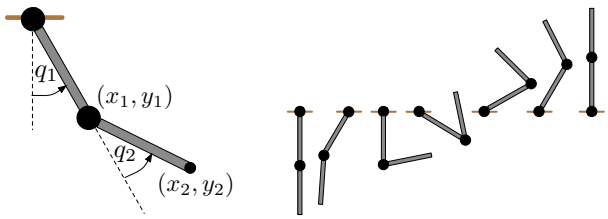
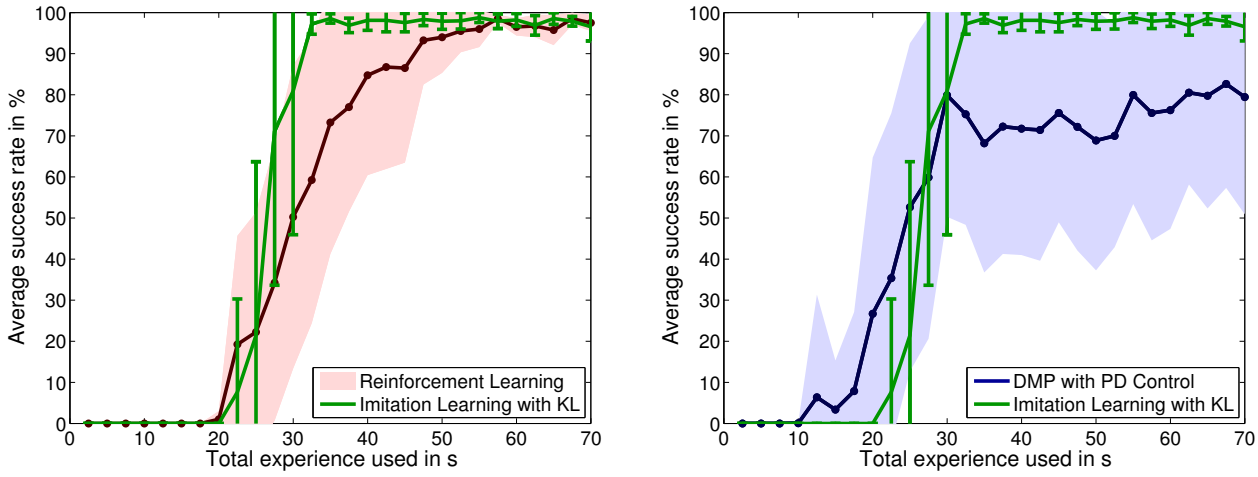


Figure 5: Parametrization of the double pendulum and a sequence of configurations during the upswing task.



(a) Comparison to Reinforcement Learning

(b) Comparison to Imitation Learning with DMPs

Figure 7: Average success rate for solving the double pendulum swing-up task as a function of required data the robot effectively used for learning. The green error bars in (a) and (b) show the performance of the model-based IL approach described in Section 4. The shaded red graph in (a) shows the success rate of an RL controller with the hand-crafted cost function from Equation (28). The shaded blue graph in (b) shows the DMP-based imitation learning controller from Equation (29). All learned models were initialized with a random rollout. Therefore, they start at 2.5 s.

5.1.1 Comparison to Reinforcement Learning and Imitation Learning

To qualitatively evaluate learning speed, we compared our proposed model-based IL approach with RL approaches that learn the double-pendulum swing-up task. Unlike our IL approach, which matches trajectory distributions by minimizing their KL divergence, RL requires a hand-crafted cost function, where we chose the common cost function

$$c(\boldsymbol{x}) = 1 - \exp(-||\boldsymbol{x} - \boldsymbol{x}_{\text{target}}||^2/8) \tag{28}$$

with the target state $\boldsymbol{x}_{\text{target}}$ in task space. As RL algorithm, we used the PILCO policy search framework [13], which allows an adequate comparison of the learning speed since it also learns a probabilistic forward model. The average success rate as a function of required data is visualized in Figure 7. A run was considered successful, when the double pendulum performed the swing-up and balanced in the inverted position. The success rate is given in percent and averaged over ten independent experiments of the algorithms. In each experiment, the model was initialized with a different random rollout. The shaded red graph represents PILCO's learning speed and reaches a success rate of $95\,\%$ after about $50\,\text{s}$ of robot interactions. The performance of the model-based IL algorithm with the KL divergence is visualized as the green graph and reaches a similar success rate after about $33\,\text{s}$ only. This performance boost can be explained by the fact that the RL method with the hand-crafted cost function initially needs to explore good trajectories that lead to the desired configuration. Our imitation learning algorithm instead exploits information about the desired trajectories through the expert's trajectory distribution that pushes the exploration towards states of these expert trajectories. The presence of demonstrations that guide the learning process can be considered a kind of natural reward shaping [26] that speeds up learning.

For evaluating the robustness of our IL controller, we compared our approach to another model-based imitation learning method. In particular, we used a learned inverse model combined with a PD controller, such that our policy was given by

$$\pi(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = f_{\text{inv}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{q}_{t+1}^{\text{exp}}, \dot{\boldsymbol{q}}_{t+1}^{\text{exp}}, \ddot{\boldsymbol{q}}_{t+1}^{\text{exp}}) + \boldsymbol{K}_p(\boldsymbol{q}_t^{\text{exp}} - \boldsymbol{q}_t) + \boldsymbol{K}_d(\dot{\boldsymbol{q}}_t^{\text{exp}} - \dot{\boldsymbol{q}}_t), \tag{29}$$

where we used dynamic movement primitives (DMPs) for representing the expert movement [35, 36]. The inverse model was learned incrementally with GPs similar to our forward model as described in Section 3.2.

As inputs of the GP model we used the current joint position and velocity $[\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t]$ and the desired joint position, velocity and acceleration of the next state $[\boldsymbol{q}_{t+1}^{\text{exp}}, \dot{\boldsymbol{q}}_{t+1}^{\text{exp}}, \ddot{\boldsymbol{q}}_{t+1}^{\text{exp}}]$, which were given by the DMP. The outputs of the model were feedforward torques, which are the left-hand-side in Equation (29). We combined this feedforward part with a feedback controller where we selected $\boldsymbol{K}_p = 0.5\boldsymbol{I}$ and $\boldsymbol{K}_d = 0.25\boldsymbol{I}$ as suited. The success rate of this method is visualized as the blue shaded graph in Figure 7. In each run for estimating the success rate, the start position of both approaches was sampled from the same probability distribution $p(\boldsymbol{x}_0)$ to test the robustness of the controllers. It can be seen that the DMP-based method was able to solve the task and reached an average success rate of about $80\,\%$. This method learned initially faster than our IL approach since the PD controller pushed the robot into the direction of the expert demonstrations. However, the DMP-based approach did not reach a high success rate, due to the variations in the initial position. Our imitation learning method with the KL divergence shows a higher robustness and solved the task with an average success rate of about $95\,\%$.

These overall results show that the KL divergence is an appropriate measure for matching trajectory distributions in the context of IL, which leads to a fast learning of robust policies.

5.1.2 Addressing the Correspondence Problem

Our approach is more robust to the correspondence problem than other imitation learning methods (e.g., behavioral cloning) because it learns robot-specific controllers. This property gives us the ability to learn controllers for cases where the robot and the teacher do not possess the same properties (i.e., changing the weight at the robot's end effector). For example, in our experiments with the double pendulum, we applied our IL method to the double-pendulum swing-up task, where the mass values of the second link were different during demonstrations and the robot's execution. The expert trajectories were created with a mass of $0.5\,\text{kg}$ of the second link. We tested our IL approach in cases where the robot's second link masses ranged between $0.15\,\text{kg}$ and $1\,\text{kg}$. In such a case, classical behavioral cloning would fail because the recorded state-actions pairs do not resemble the robot behavior with the changed attributes. Our imitation learning approach, however, could still learn a controller as we search for a controller that takes the different attributes during learning into account. Our approach successfully performed the task in the range between $0.15\,\text{kg}$ and $0.57\,\text{kg}$ for the second link. Learning did not succeed for mass values above $0.57\,\text{kg}$, due to the torque limits. Thus, our approach is not robust to all kinds of correspondence problems, particularly if the required control commands for imitating the teacher exceed the torque limits of the robot. In this case, we cannot imitate the teacher. However, we may sometimes still find good solutions.

5.2 Kinesthetic Teaching with a BioRob

In this section, we evaluate the performance of our imitation learning approach on a real robot. We used the biomechanically-inspired compliant BioRob$^{\text{TM}}$ [22] to learn a fast ball-hitting movement that we demonstrated via kinesthetic teaching. We describe first the robot hardware and the experimental set-up, and afterward we detail model and controller learning.

5.2.1 Hardware Description

The BioRob$^{\text{TM}}$ (see Figure 1) is a compliant, light-weight robotic arm, capable of achieving high accelerations. Its design tries to place the servo motors close to the torso, minimizing the inertia of the links and enable the end-effector to move with high velocities. Experimental results have shown Cartesian velocities of the end-effector of up to $6.88\,\text{m/s}$ [21]. The BioRob X4 is equipped with an end-effector module that increases the total number of degree of freedom to five. The torque is transferred from the motor to the joints via a system of pulleys, drive cables, and springs, which, in the biologically-inspired context, represent tendons and their elasticity. In terms of safety, decoupling the joint and motor inertia protects the items in the robot's workspace and the motor gearboxes in the event of collisions. While the BioRob's design has advantages over traditional approaches, it has the disadvantage that controlling such a compliant system is a highly challenging task.

Classical control approaches that consider only the rigid body dynamics of the system are unrealistic for controlling the robot as they omit the cable-driven properties, such as the elasticity of the tendons, the cable

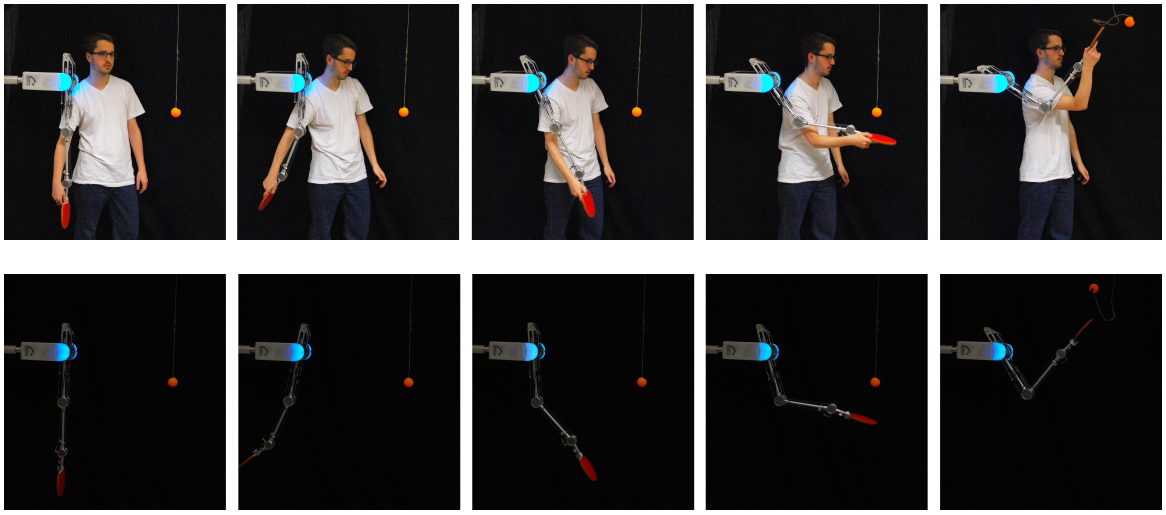


Figure 8: The upper photo series shows a kinesthetic demonstration by an expert and the lower photos show the learned movement after the fourth iteration of our approach.

slacking effects, stiction, and the energy stored in the cable springs. Linear control approaches suffer even more form the actuator dynamics. As a result, both forward and inverse rigid-body dynamics models are not sufficiently accurate for classical control, and the robot fails to follow desired trajectories not even approximately. Moreover, if the control torques are not sufficiently smooth, oscillations close to the eigen-frequency occur. During the oscillations, the motors hold the same position, while the joints, due to the kinematic decoupling from the motors, oscillate. These oscillations differ from the classical under-damped control systems, and, thus, damping them is a non-trivial task.

5.2.2 Task Setup

We attached a table tennis racket to the end-effector of the robot and attached a ball to a string hanging down from the ceiling, see Figure 8. The shape of the racket alongside with the high velocities produces a significant amount of drag, which is hard to model accurately, leading to substantial errors in parametric models. Thus, learning a non-parametric GP forward model that extracts useful information from data and, subsequently, learning control policies for solving the task, is particularly promising for this compliant tendon-driven robot.

We controlled three joints of the BioRob$^{\text{TM}}$ for performing the task. The state $\boldsymbol{x} \in \mathbb{R}^6$ was given by three joint positions and velocities of the robot; the actions $\boldsymbol{u} \in \mathbb{R}^3$ were given by the corresponding motor torques. The applied motor commands to the robot were the outcome of the policy in Equation (27) without any feedback component. We provided three demonstrations of the task via kinesthetic teaching, as shown in Figure 8, to create a distribution over expert trajectories. Therefore, we took the robot by the hand and recorded the system states. The task was first to move back and then to perform a fast up movement to hit the ball, see Figure 8.

5.2.3 Model Learning

An important parameter when learning models is the sampling frequency $f_s = 1/\Delta T$ where ΔT is the time difference between $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$ in our learned forward model, see Equations (2)–(4). High frequencies result in increased computational time as the number of time steps for a given prediction horizon increases. Moreover, changes in succeeding states can be too insignificant to learn a robust model because of a low signal-to-noise ratio: Small changes increase the risk that important information about the underlying function is filtered out.

For finding an appropriate sampling frequency f_s, we used k-fold cross-validation with the log-likelihood of our GP predictions as performance measure. We divided the recorded data into $k = 5$ training/test folds and computed for each fold the predictive log-likelihood with different f_s values. The log-likelihood for one fold is

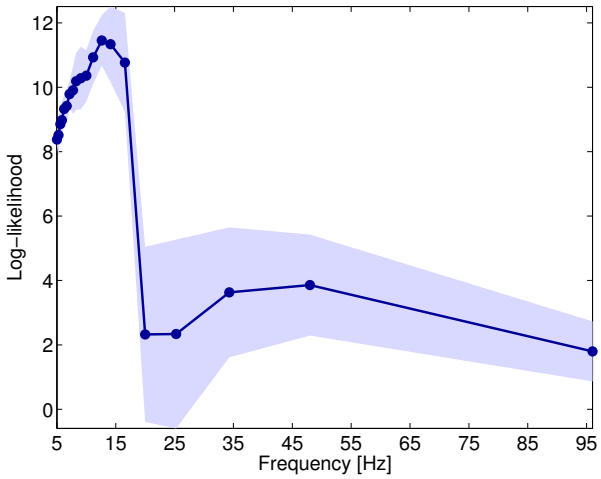
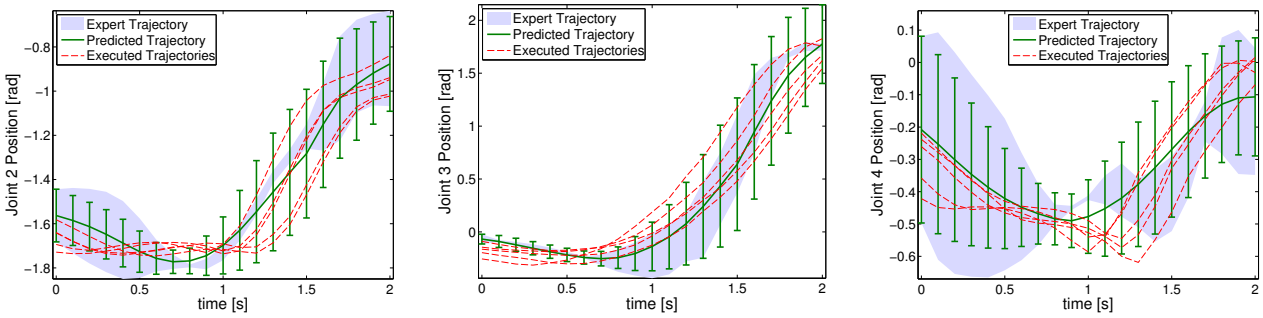


Figure 9: Cross-validation of the frequency value f_s in the BioRob experiment. As a performance measure of the model we use the predictive log-likelihood. High sampling frequencies increase the risk of overfitting, i.e., important information about the underlying function is filtered out since the signal-to-noise ratio is too small. Small sampling frequencies increase the risk of underfitting, i.e., important variability of the underlying function is not captured due to under-sampling. The ideal sampling frequency of the data is at about 10 Hz.



(a) Position of Joint 2. (b) Position of Joint 3. (c) Position of Joint 4.

Figure 10: Results after learning the imitation of a task with the BioRob$^{\text{TM}}$ from kinesthetic teaching. The figures above show the distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over expert trajectories (shaded blue area) and the distribution $p(\boldsymbol{\tau}^{\pi})$ over predicted trajectories from the learned forward model (green error bars). Both are plotted with two times the standard deviation. The red dashed lines show some executed trajectories of the robot where we applied the learned policy. There start state was sampled from the initial distribution $p(\boldsymbol{x}_0)$.

defined as

$$\log p(\boldsymbol{y}_i|\boldsymbol{X}, \boldsymbol{y}_{-i}) = -\tfrac{1}{2}\log|\boldsymbol{\Sigma}_i| - \tfrac{D}{2}\log(2\pi) - \tfrac{1}{2}(\boldsymbol{y}_i - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y}_i - \boldsymbol{\mu}_i). \tag{30}$$

Here, $\boldsymbol{y}_{-i}$ denotes the training set without the test values $\boldsymbol{y}_i$ of the current fold, D is the number of test values and $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the predicted mean and variance of the test inputs $\boldsymbol{x}_i$ according to Equations (6)–(7), respectively. Figure 9 shows the averaged log-likelihood over different frequencies f_s. These results show that a sampling frequency f_s of around 10 Hz is most suited for model learning. Higher f_s values reach a lower predictive log-likelihood, which can be explained either by the fact that either they fit to the measurement noise leading to overfitting or the signal-to-noise ratio is very low.

5.2.4 Controller Learning

For learning a BioRob controller we used the RBF network from Equation (27) with 80 basis functions, resulting in 738 policy parameters. According to the cross-validation results from Section 5.2.3, we selected a sampling frequency of 10 Hz as optimal for our GP forward model. We recorded state-action pairs to initialize the GP forward model. Therefore, we executed simple movements with a PD controller at robot configurations

around the start distribution $p(\boldsymbol{x}_0)$ of the task. These data points corresponded to a total experience of $6\,\mathrm{s}$. Our probabilistic IL approach based on trajectory matching led to rapid learning. After the second attempt, the robot was already able to hit the ball and to do a movement similar to the teacher's. After the fourth trial, the robot solved the task and could imitate the demonstrations reliable (see Figure 10).

The predicted distribution $p(\boldsymbol{\tau}^\pi)$ over robot trajectories, the expert distribution $p(\boldsymbol{\tau}^{\text{exp}})$ over demonstrated trajectories, and some executed trajectories after four learning iterations of our proposed IL approach are visualized in Figure 10. The figure shows the positions of the three joints that were used for the task. The trajectory prediction of the GP is shown as green error bars. The blue shaded graph is the expert trajectory. Some executed trajectories of the robot where we applied the learned policy are shown as red dashed lines. The robot was able to imitate the demonstrations in a robust manner from different starting positions, using a total of less than $30\,\mathrm{s}$ of data to learn both an accurate forward model and a robust controller.

5.2.5 Generalization to Multiple Targets

The policy that the robot learned in the previous experiments was able to imitate a single task without any generalization abilities. One of the key challenges in imitation learning is the generalization to more complex tasks, where a policy needs to be adaptable to changes in the environment (e.g., to an object position).

To generalize the policy to different target positions we incorporated our IL framework into the multi-target scenario proposed in [12]. The key idea is to directly parametrize the policy π by the target variables $\boldsymbol{\eta}$. The policy $\pi(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\eta})$ is trained on a small set of training targets $\boldsymbol{\eta}_i^{\text{train}}$, e.g., the orange balls in Figure 11(a). In the test phase, the policy generalizes to test targets $\boldsymbol{\eta}_i^{\text{test}}$ that are previously unseen but related to the training targets. Note that these test targets do not have to be inside the convex hull of the training targets [12].

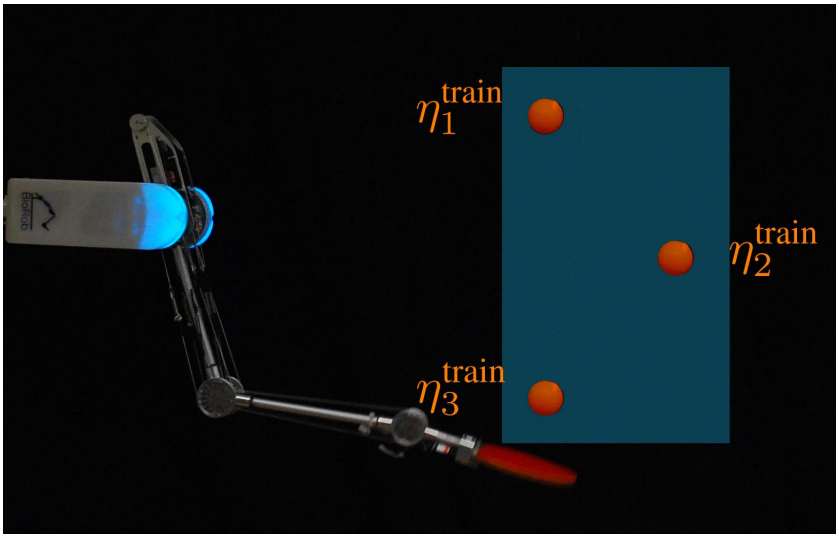
We define the multiple target objective function as

$$J_{\text{MT}}^\pi(\boldsymbol{\theta}, \boldsymbol{\eta}) \approx \frac{1}{M} \sum_{i=1}^{M} J_{\text{IL}}^\pi \left(\boldsymbol{\theta}, \boldsymbol{\eta}_i^{\text{train}}\right) = \frac{1}{M} \sum_{i=1}^{M} \text{KL}\left(p(\boldsymbol{\tau}_i^{\text{exp}}) || p(\boldsymbol{\tau}^\pi | \boldsymbol{\eta}_i^{\text{train}})\right) , \tag{31}$$

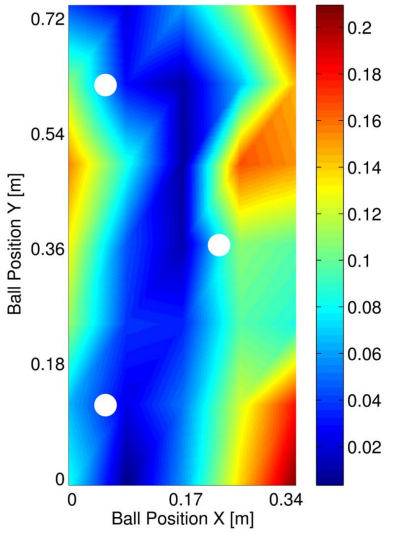
where we sum the objective function J_{IL}^π from Equation (1) over M training targets $\boldsymbol{\eta}_i^{\text{train}}$. With the objective function in Equation (31) the policy parameters are learned jointly for all targets. In the context of our imitation-learning set-up, we demonstrated multiple trajectories to estimate the corresponding expert trajectory distribution $p(\boldsymbol{\tau}_i^{\text{exp}})$ for each training target. Optimizing Equation (31) with Algorithm 1 allows the robot to learn a single policy, explicitly parametrized by the target location $\boldsymbol{\eta}$.

We evaluated the generalization ability of our approach by extending the experiments from Section 5.2.4 to variable ball positions in a 2D-plane, see Figure 11(a). A target was represented as a two-dimensional vector $\boldsymbol{\eta} \in \mathbb{R}^2$ corresponding to the ball position in Cartesian coordinates in an arbitrary reference frame within the hitting plane. As training targets $\boldsymbol{\eta}_j^{\text{train}}$, we defined hitting movements for three different ball positions (see Figure 11(a)). For each training target, an expert demonstrated two hitting movements via kinesthetic teaching. Our goal was to learn a policy that a) learns to imitate three distinct expert demonstrations, and b) generalizes from demonstrated behaviors to targets that were not demonstrated. In particular, these test targets were to hit balls in a larger region around the training locations, indicated by the blue box in Figure 11(a). The controller automatically learned the implicit similarity of the targets. As the policy parametrization, we used the RBF network in Equation (27) with 250 Gaussian basis functions, resulting in 2774 policy parameters $\boldsymbol{\theta}$. For a detailed description of this approach, we refer to [12].

Figure 11(b) shows the performance results as heatmap after 15 iterations of Algorithm 1. The evaluation measure was the distance between the ball position and the center of the table-tennis racket. We computed this error in a regular 7x5 grid of the blue area in Figure 11(b). The distances in the blue and cyan areas were sufficient to successfully hit the ball. We conclude that our approach successfully generalized demonstrations to new targets.



(a) Set-up of the multiple target imitation learning experiments.



(b) Evaluation of the multiple target imitation learning experiments.

Figure 11: (a) Set-up for the imitation learning experiments. The orange balls represent the three training targets $\boldsymbol{\eta}_i^{\text{train}}$. The blue rectangle indicates the regions of the test targets $\boldsymbol{\eta}_j^{\text{test}}$ for our learned controller to which we want to generalize. (b) Evaluation of the imitation learning experiments with the BioRob$^{\text{TM}}$. The three white discs show the training target locations. The color encodes the minimum distance between the ball position and the trajectory of the center of the table-tennis racket. In the blue and cyan areas, the robot successfully hit the table tennis ball.

6 Conclusion

In this paper, we have presented a probabilistic model-based imitation learning approach that enables robots to acquire new tricks through teacher demonstrations. The three key components of our approach are: 1) probabilistic modeling of both the robot's dynamics and the teacher's demonstrations allows us to take uncertainty appropriately into account; 2) mental rehearsal of the current controller with predictions of distributions over plausible trajectories guarantees data efficient learning; 3) searching for robot-specific controllers that match the robot trajectory with the expert trajectory enables us to use demonstration methods that do not record the actions of the teacher. We have shown that matching trajectory distributions by means of minimizing their Kullback-Leibler divergence is equivalent to a reinforcement learning problem with an induced time-varying immediate cost function. We addressed the correspondence problem by learning robot-specific controllers. We demonstrated that differences in the dynamics between teacher and robot do not disrupt learning as long as the kinematic and dynamics restrictions of the robot are not passed. Our experimental results have shown that our approach learns faster than a reinforcement learning with hand-crafted cost functions. Additionally, the comparison to the model-based imitation learning method with DMPs showed that our approach has a high robustness to changes in the task set-up. Furthermore, we demonstrated the applicability of our approach to real robots, where we used a compliant robot to imitate demonstrations provided by kinesthetic teaching in a fast and robust manner. We also showed that our approach applies to more complex imitation tasks with multiple targets.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement #270327.

References

[1] P. Abbeel, D. Dolgov, A.Y. Ng, and S. Thrun. Apprenticeship Learning for Motion Planning with Application to Parking Lot Navigation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2008.

[2] B.D. Argall, S. Chernova, M. Veloso, and B. Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[3] C.G. Atkeson and J. C. Santamaría. A Comparison of Direct and Model-Based Reinforcement Learning. In *Proceedings of the International Conference on Robotics and Automation*, 1997.

[4] C.G. Atkeson and S. Schaal. Robot Learning from Demonstration. In *Proceedings of the International Conference on Machine Learning*, 1997.

[5] J.A. Bagnell and J.G. Schneider. Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods. In *Proceedings of the International Conference on Robotics and Automation*, 2001.

[6] M. Bain and C. Sammut. A Framework for Behavioural Cloning. *Machine Intelligence*, 15:103–129, 1999.

[7] H. Ben Amor, E. Berger, D. Vogt, and B. Jung. Kinesthetic Bootstrapping: Teaching Motor Skills to Humanoid Robots through Physical Interaction. *KI 2009: Advances in Artificial Intelligence*, pages 492–499, 2009.

[8] G. Biggs and B. Macdonald. A Survey of Robot Programming Systems. In *Proceedings of the Australian Conference on Robotics and Automation*, pages 1–3, 2003.

[9] A. Boularias, J. Kober, and J. Peters. Relative Entropy Inverse Reinforcement Learning. *In Proceedings of AISTATS*, 2011.

[10] S. Calinon, F. Guenter, and A. Billard. On Learning, Representing, and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(2):286–298, 2007.

[11] L. Carlone, J. Du, M.K. Ng, B. Bona, and M. Indri. An Application of Kullback-Leibler Divergence to Active SLAM and Exploration with Particle Filters. In *Proceedings of International Conference on Intelligent Robots and Systems*, 2010.

[12] M.P. Deisenroth and D. Fox. Multiple-Target Reinforcement Learning with a Single Policy. *In ICML 2011 Workshop on Planning and Acting with Uncertain Models*, 2011.

[13] M.P. Deisenroth and C.E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the International Conference on Machine Learning*, 2011.

[14] M.P. Deisenroth, C.E. Rasmussen, and D. Fox. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Proceedings of Robotics: Science & Systems*, 2011.

[15] K. Doya. Reinforcement Learning in Continuous Time and Space. *Neural Computation*, 12(1):219–245, 2000.

[16] P. Englert, A. Paraschos, J. Peters, and M.P. Deisenroth. Model-based Imitation Learning by Probabilistic Trajectory Matching. *Proceedings of the International Conference on Robotics and Automation*, 2013.

[17] H. Hoffmann, P. Pastor, D. Park, and S. Schaal. Biologically-inspired Dynamical Systems for Movement Generation: Automatic Real-time Goal Adaptation and Obstacle Avoidance. In *Proceedings of International Conference on Robotics and Automation*, 2009.

[18] T. Inamura, H. Tanie, and Y. Nakamura. From Stochastic Motion Generation and Recognition to Geometric Symbol Development and Manipulation. In *Proceedings of International Conference on Humanoid Robots*, 2003.

[19] J. Kober and J. Peters. Imitation and Reinforcement Learning. *IEEE Robotics & Automation Magazine*, 17(2):55–62, 2010.

[20] J.Z. Kolter, P. Abbeel, and A.Y. Ng. Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion. *Advances in Neural Information Processing Systems*, 2008.

[21] T. Lens. *Physical Human-Robot Interaction with a Lightweight, Elastic Tendon Driven Robotic Arm: Modeling, Control, and Safety Analysis*. PhD thesis, TU Darmstadt, Department of Computer Science, 2012.

[22] T. Lens, J. Kunz, O. von Stryk, C. Trommer, and A. Karguth. BioRob-Arm: A Quickly Deployable and Intrinsically Safe, Light-Weight Robot Arm for Service Robotics Applications. *International Symposium on Robotics*, 2010.

[23] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to Select and Generalize Striking Movements in Robot Table Tennis. *International Journal of Robotics Research*, 2013.

[24] Y. Nakamura, W. Takano, and K. Yamane. Mimetic Communication Theory for Humanoid Robots Interacting with Humans. *Robotics Research*, pages 128–139, 2007.

[25] C.L. Nehaniv and K. Dautenhahn. The Correspondence Problem. *In Imitation in Animals and Artifacts*, page 41, 2002.

[26] A.Y. Ng, D. Harada, and S. Russell. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the International Conference on Machine Learning*, 1999.

[27] A.Y. Ng and M. Jordan. PEGASUS: A Policy Search Method for Large MDPs and POMDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2000.

[28] A.Y. Ng, H.J. Kim, M.I. Jordan, and S. Sastry. Autonomous Helicopter Flight Via Reinforcement Learning. In *Proceedings of the International Symposium on Experimental Robotics*, 2003.

[29] A.Y. Ng and S. Russell. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*, 2000.

[30] J. Peters, K. Mülling, and Y. Altun. Relative Entropy Policy Search. In *Proceedings of the Conference on Artificial Intelligence*, 2010.

[31] D.A. Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. 1989.

[32] J. Quiñonero-Candela, A. Girard, J. Larsen, and C.E. Rasmussen. Propagation of Uncertainty in Bayesian Kernel Models—Application to Multiple-Step Ahead Forecasting. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2:701–704, 2003.

[33] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[34] H. Sakoe and S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[35] S. Schaal, A. Ijspeert, and A. Billard. Computational Approaches to Motor Learning by Imitation. *Philosophical Transactions of the Royal Society of London*, 358:537–547, 2003.

[36] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning Movement Primitives. *Robotics Research*, pages 561–572, 2005.

[37] J.G. Schneider. Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning. In *Advances in Neural Information Processing Systems*. 1997.

[38] K. Solomon. *Information Theory and Statistics*. Wiley, New York, 1959.

[39] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley New Jersey, 2006.

[40] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[41] A. Ude, C.G. Atkeson, and M. Riley. Programming Full-Body Movements for Humanoid Robots by Observation. *Robotics and Autonomous Systems*, 47(2):93–108, 2004.

[42] D.M. Wolpert, Z. Ghahramani, and M.I. Jordan. An Internal Model for Sensorimotor Integration. *Science*, 269:1880–1882, 1995.

[43] R. Zöllner, T. Asfour, and R. Dillmann. Programming by Demonstration: Dual-Arm Manipulation Tasks for Humanoid Robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2004.